EE 474 Lab 3: LKMs and Shift Registers

University of Washington
by Brad Marquez, Joseph Rothlin, Aigerim Shintemirova
EE 474

May 13, 2016

1. Abstract

The main purpose of this lab is to introduce and explore the concepts of Kernel mode programming while optimizing the code done in the previous lab. First, an 8-bit shift register is designed in order to reduce the number of GPIO pins used. Second, a Kernel mode driver, LKM, is built to interface with the LCD screen on the board. From kernel mode, we developed a LKM that allows us to open, close, read, and write to the LCD.

2. Development:

1. Connecting to the Board

Establishing the proper connection to the board is essential to accomplish the goal of this lab. It is done using SSH networking protocol commands. The command used was ssh root@192.168.7.2. Once, the connection is established, programs could be loaded onto the board.

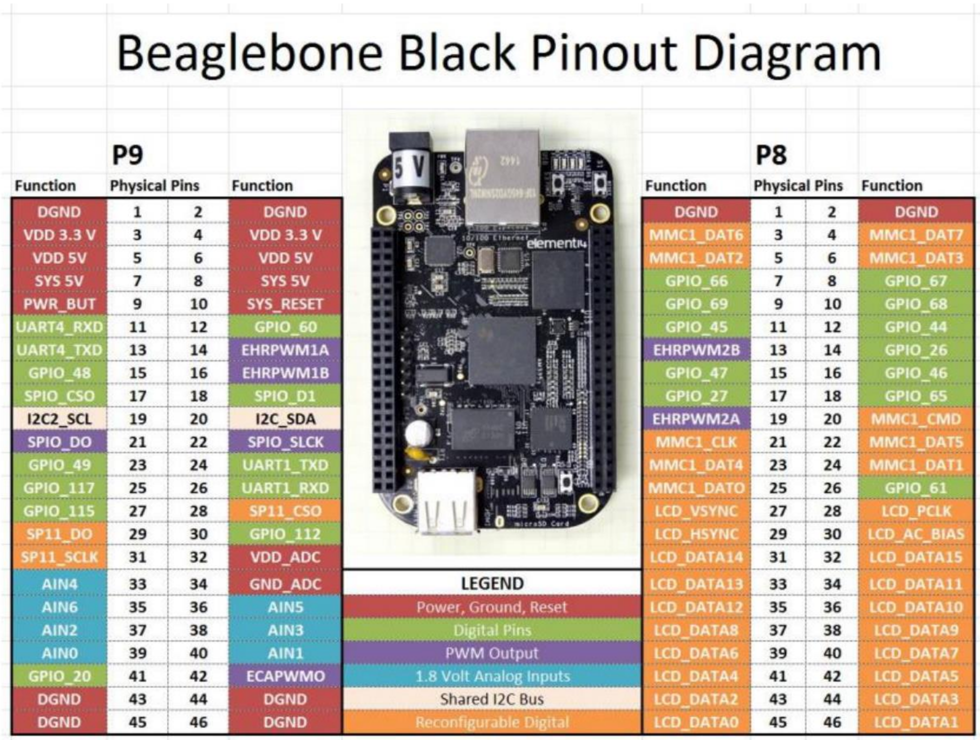

Figure 1. Pinout for the Beaglebone black board. Notice that the export reference to the GPIO pins is denoted by the number within the Function name. For example, the number used to denote Physical Pin 15, P9 is 48 (GPIO_48).

2. Connecting the hardware

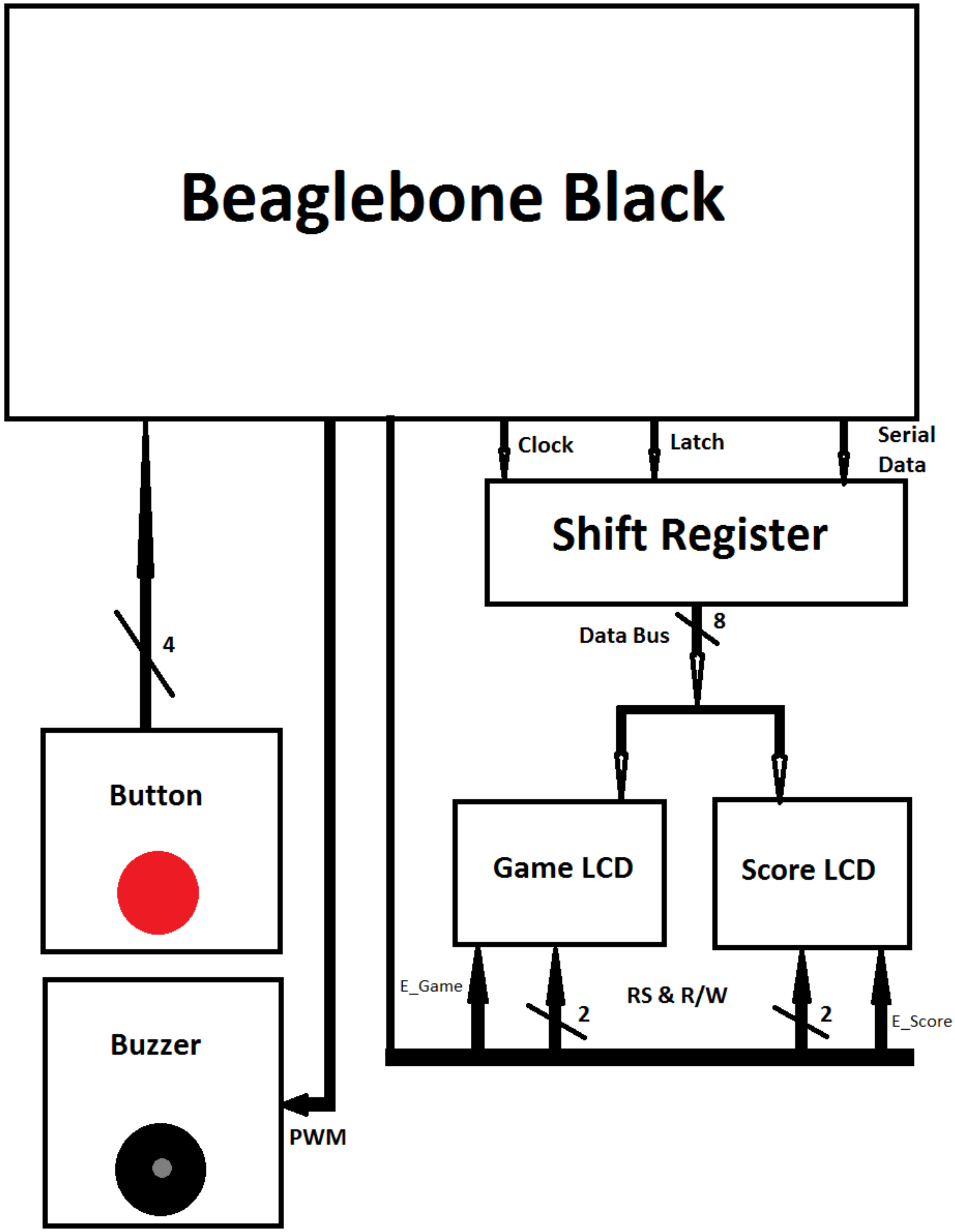

Figure 2. This is a block diagram that represents the interface between the LCD with the Beaglebone Black. Note that the buzzer is not defined within the LKM of either the button or the LCD drivers. The buzzer is defined in user space within the button Hero code that we made.

Table 1: The Pin Description for the LCD pins and Beaglebone Black pins

Pin No.	**Symbol**	**External Connection**	**Function Description**	**Attached Beaglebone Pin**
1	VSS	Power Supply	Ground	DGND
2	VDD	Power Supply	Supplies Voltage for logical operations (+5 V)	SYS 5V
3	V0	Adj Power Supply	Power supply for screen contrast	N/A
4	RS	MPU	Register select signal 0: Command, 1: Data	GPIO_PIN_68
5	R/W	MPU	Read/Write select signal 0: Write, 1: Read	GPIO_PIN_44
6	E_GAME	MPU	Operation enable signal. Sends on falling edge.	GPIO_PIN_60
7	E_SCORE	MPU	Operation enable signal. Sends on falling edge.	GPIO_PIN_26
8	DATA_	MPU	Low order bi-directional data bus lines	GPIO_PIN_45
9	LATCH_	MPU	Latch signal input	GPIO_PIN_47
10	CLOCK_	MPU	Clock signal input	GPIO_PIN_67
11	BUZZER_	MPU	PWM buzzer input	GPIO_PWM_14

2. Transferring cross-compiled files to the board

Files are transferred and cross-compiled directly to the board using the given makefile that compiles C code into LKM. We were able to set the makefile so that it compiles both the lcd_driver.o and button_driver.o at the same time. The compiler used in the makefile is the arm-linux-gnueabihf- compiler.

Table 2. The targets of the Makefile and their individual effects

Target	Function Description
default	Creates Kernel mode driver for the LCD screens and button
clean	Removes the transferred file from the Beaglebone

3. Initializing the LCD

Before being able to use the LCD, the LCD must be initialized by sending a specific sequence of instructions after the LCD has been power ON. Our group decided to use the 8-Bit LCD Interface in order to access more characters. The instruction sequence is shown below in Figure 3.

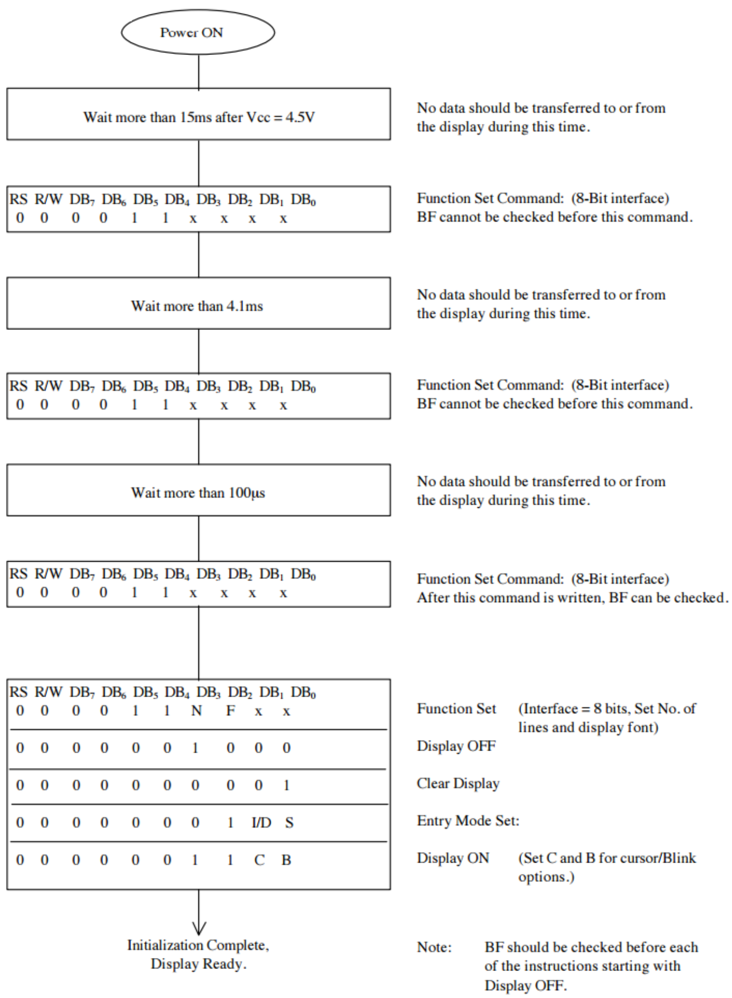

Figure 3. The instruction sequence used to initialize the NHD-0216BZ-FL-YBW LCD display to 8-bit. Note that we initialized one LCD with an 8-bit interface, 2 lines, 5x8 dot font, increment cursor mode, no display shift, cursor OFF, and blink OFF. The other

LCD was initialized to an 8-bit interface, 1 line, 5x10 dto font, decrement cursor mode, no display shift, cursor ON, and blink ON.

1. **Shift Register**
   In order to reduce the number of GPIO pins used to interface with the LCD display we designed an 8-bit shift register. As a result, only three GPIO pins, clock, latch, and data, were used to replace DB7...DB0 for two LCD screens.
2. **Kernel Mode**
   In the second part of the lab, we designed a kernel mode device driver for the LCD screens and a button used to navigate the Button Hero game. It can execute any instruction on the CPU without waiting and it has access to all of the memory addresses.

3. **Button Hero Game**
   Lastly, we designed a Button Hero game in user space mode. The game is executed on two LCD screens and the control is done using a button. A random sequence of five direction characters, up, down, left, right, and press, is generated and then the button is used to mimic the character on the screen. As the game progresses, the first LCD screen is used to play the game and the second one displays the score and the misses.

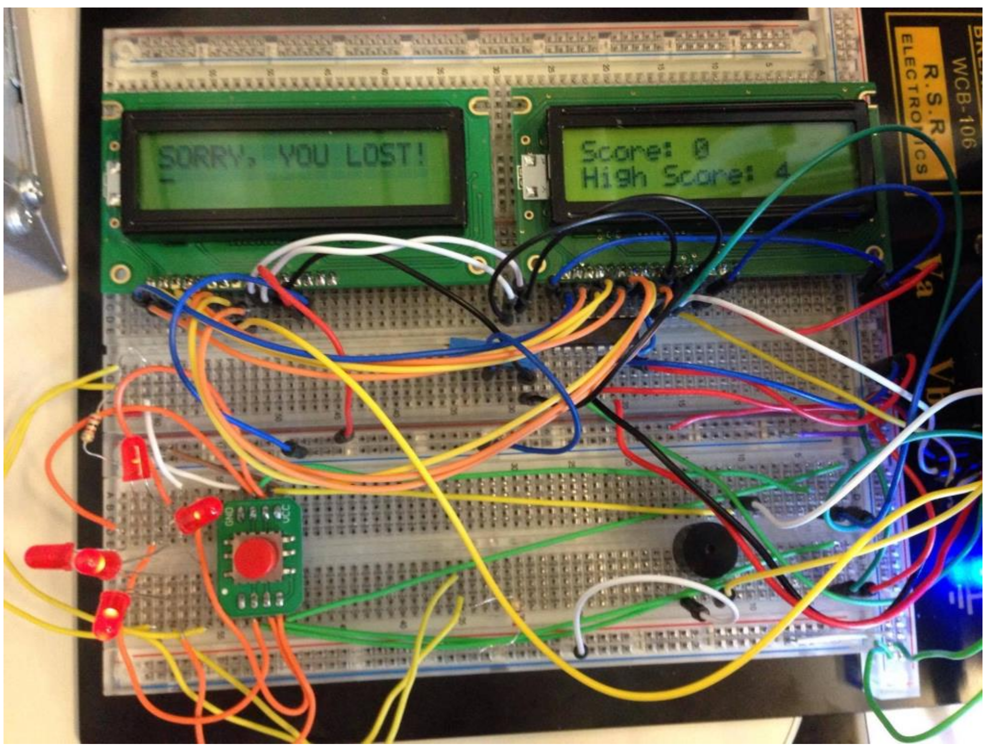

Figure 4. Overall system design with two LCD screens, five LEDs, button, and a buzzer with the Button Hero game results on the screens.

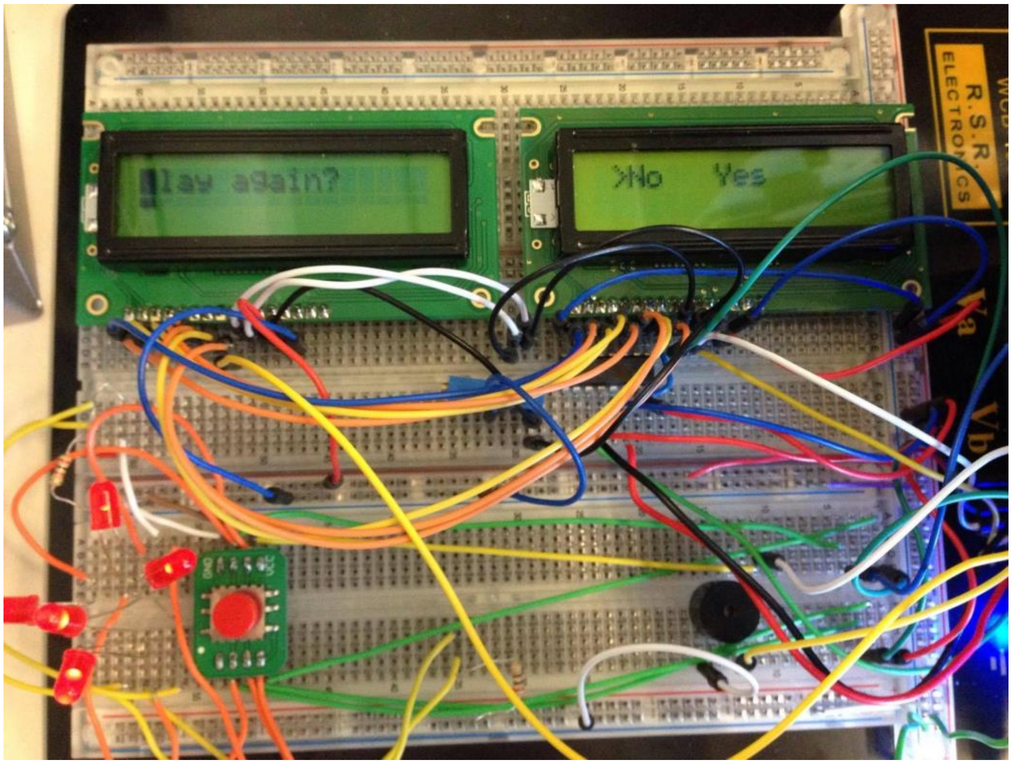

Figure 5. LCD screens with the main menu after the game is played

3. Discussion

While working on this lab we were able to explore the design of kernel mode device drivers and integrate it with the game in the user space.

4. Conclusion

The main purpose of this lab was to design a shift register to decrease the number of GPIO pins used. As a result, instead of using eight GPIO pins for each of the two 8-bit LCD screens, only three were used: clock, latch, and data.
Then, kernel mode device drivers were created to interface with the LCD screens and a button for the game.